06

Haga lo que hicimos en aula: Criteria API

Cuando queremos realizar consultas con múltiples parámetros nos encontramos con el problemas que todos ellos deben ser obligatorios, de lo contrario consultaria elementos nulos en tabla. Para evitar este error tenemos que usar parámetros dinámicos que nos permiten realizar consultas con múltiples parámetros y en caso de que alguno de estos sea nulo la consulta simplemente ignorará este parámetro y realizará la consulta con los parámetros existentes. Y en caso de no existir ningún parámetro, realizará la consulta de todos los elementos en la tabla.

```
public List<Producto> consultarPorParametros(String nombre, Bi
    StringBuilder jpql=new StringBuilder("SELECT p FROM Produc

    if(nombre!=null && !nombre.trim().isEmpty()) {
        jpql.append("AND p.nombre=:nombre ");
    }
    if(precio!=null && !precio.equals(new BigDecimal(0))) {
        jpql.append("AND p.precio=:precio ");
    }
    if(fecha!=null) {
        jpql.append("AND p.fechaDeRegistro=:fecha");
    }
    TypedQuery<Producto> query = em.createQuery(jpql.toString(
    if(nombre!=null && !nombre.trim().isEmpty()) {
        query.setParameter("nombre", nombre);
    }
    if(precio!=null && !precio.equals(new BigDecimal(0))) {
        query.setParameter("precio", precio);
    }
```

```
    if(fecha!=null) {
        query.setParameter("fechaDeRegistro", fecha);
    }
    return query.getResultList();
}
```

COPIA EL CÓDIGO

- Adicionalmente podemos realizar la misma consulta dinámica utilizando la API de Criteria que es un poco más compleja y recomendamos documentarse sobre ella pero simplifica la cantidad de condiciones en nuestra aplicación.

```
public List<Producto> consultarPorParametrosConAPICriteria(Str
    CriteriaBuilder builder = em.getCriteriaBuilder();
    CriteriaQuery<Producto> query = builder.createQuery(Produc
    Root<Producto> from = query.from(Producto.class);

    Predicate filtro = builder.and();
    if(nombre!=null && !nombre.trim().isEmpty()) {
        filtro=builder.and(filtro,builder.equal(from.get("nombr
    }
    if(precio!=null && !precio.equals(new BigDecimal(0))) {
        filtro=builder.and(filtro,builder.equal(from.get("preci
    }
    if(fecha!=null) {
        filtro=builder.and(filtro,builder.equal(from.get("fecha
    }
    query=query.where(filtro);
    return em.createQuery(query).getResultList();
}
public class PruebaDeParametros {
    public static void main(String[] args) {
```

```
        cargarBancoDeDatos();

        EntityManager em = JPAUtils.getEntityManager();
        ProductoDao productoDao =new ProductoDao(em);

        List<Producto> resultado = productoDao.consultarPorPar

        System.out.println(resultado.get(0).getDescripcion());

EntityManager em = JPAUtils.getEntityManager();
        ProductoDao productoDao =new ProductoDao(em);

        List<Producto> resultado = productoDao.consultarPorPar

        System.out.println(resultado.get(0).getDescripcion());

    }

    private static void cargarBancoDeDatos() {
        Categoria celulares = new Categoria("CELULARES");
        Categoria videoJuegos = new Categoria("VIDEO_JUEGOS");
        Categoria electronicos = new Categoria("ELECTRONICOS")

        Producto celular = new Producto("X", "producto nuevo",
        Producto videoJuego = new Producto("FIFA", "2000", new
        Producto memoria = new Producto("memoria ram", "30 GB"

        EntityManager em = JPAUtils.getEntityManager();
        ProductoDao productoDao = new ProductoDao(em);
        CategoriaDao categoriaDao = new CategoriaDao(em);

        em.getTransaction().begin();
```

```
        categoriaDao.guardar(celulares);
        categoriaDao.guardar(videoJuegos);
        categoriaDao.guardar(electronicos);

        productoDao.guardar(celular);
        productoDao.guardar(videoJuego);
        productoDao.guardar(memoria);

        em.getTransaction().commit();
        em.close();
    }
}
```

COPIA EL CÓDIGO